University of Engineering & Management, Kolkata
University of Engineering & Management, Jaipur
Institute of Engineering & Management, Kolkata

Department of Computer Science

B.Tech in CSE (AI & ML)

COURSE STRUCTURE

Batch: 2022-2026

Semester VII (Fourth Year) Curriculum								
Sl. No	**Type of course**	**Course Code**	**Course Name**	**Hours per week**				**Credit Points**
				Lecture	**Tutorial**	**Practical**	**Sessional**	
Theory Papers								
1	Professional Core Course	PCCCS701	Compiler Design	3	0	0	0	3
2	Professional Elective Course	PECCS701	Elective-III	3	0	0	0	3
3	Open Elective Course	OECCS701	Open Elective-I	3	0	0	0	3

4	Humanities & Social Sciences including Management course	ESP(CS)701	Essential Studies for Professionals – VII (CS)	2	0	0	0	0.5
	Total			**11**	**0**	**0**	**0**	**9.5**
Practical Papers								
1	Professional Core Course	PCCCS791	Compiler Design Laboratory	0	0	4	0	2
	Total			**0**	**0**	**4**	**0**	**2**
Sessional Papers								
1	Humanities & Social Sciences including Management course	SDP781	Skill Development for Professionals - VII	0	0	0	2	0.5
2	Innovative Project	PRJCS781	Project – II	0	0	12	0	6
3	Internship	SICS781	Summer Internship - I	0	0	0	0	4
	Total			**0**	**0**	**12**	**2**	**10.5**
Mandatory Requirements								
Sl. No	**Type of course**	**Course Code**	**Course Name**	**Hours per week**				**Score/Credit/ Count**

1	Co-curricular & Extra Curricular Activities	MAR	Mandatory Additional Requirements (Score)	-	-	-	-	-
2	Honours	MOOCs	Massive Open Online Course (Credit)	-	-	-	-	-
3	Certification	IFC	Industry and Foreign Certification (Count)	-	-	-	-	-
Total				**11**	**0**	**16**	**2**	**22**

Offered Elective List

Category	Course Name	Course Code
Professional Elective - III	Computer Vision	PECCS701D
	Advanced Deep Learning	PECCS701L
Open Elective - I	Enterprise System	OECCS701A
	Entrepreneurship & Startups	OECCS701C

University of Engineering & Management, Kolkata
University of Engineering & Management, Jaipur
Institute of Engineering & Management, Kolkata
Department of Computer Science

B.Tech in CSE (AI & ML)

DETAILED SYLLABUS

Course Code- PCCCS701
Course Title – Compiler Design
Credit – 3
Category – Professional Core Course
Semester – VII
L:T:P:S – 3:0:2:0
Pre-requisite – Basic knowledge of Data Structures, Algorithms, Formal Language and Automata Theory

Course Outcomes:

CO1	Students will be able to learn the grammar specification for developing the lexical analyzer.

CO2	**Design a given parser specification design top-down and bottom-up parsers.**
CO3	**Develop syntax directed translation schemes.**
CO4	**Develop algorithms to generate code for a target machine.**

Study Material	**Coursera**	**NPTEL**	**LinkedIn Learning**	**Infosys Springboard**
Lesson Plan				

Module No.	Topic	Sub-topics	Mapping with Industry and International Academia	Lecture Hours	Corresponding Lab Assignments	Textbook Mapping
1	**Overview of compilation & Lexical Analysis**	The structure of a compiler and applications of compiler technology; Lexical analysis - The role of a lexical analyzer, specification of tokens, recognition of tokens, hand-written lexical analyzers, LEX, examples of LEX programs. Introduction to syntax analysis -Role of a parser, use of context-free grammars (CFG) in the specification of the syntax of programming languages, techniques for writing grammars for programming languages	***International Academia:*** ***https://ocw.mit.edu/courses/6-004-computation-structures-spring-2017/pages/c11/*** (MIT Open Courseware): ***AICTE-prescribed syllabus:*** ***https://drive.google.com/file/d/11CBeslrHOERE-***	8	Write a program that takes a simple code snippet (e.g., a few lines of C or a pseudo-language) as input and identifies all the tokens present, classifying them into categories like keywords, identifiers, operators, literals (integer, float, string), and	**Textbook-1** Chapters: 1 to 3 **Page No. – 1 to 158**

		(removal left recursion, etc.), non-context-free constructs in programming languages, parse trees and ambiguity, examples of programming language grammars	***XHDJSNsIHlRONmt_tYX/view?usp=sharing*** ***Industry Mapping:*** ***https://www.coursera.org/programs/iem-faculty-learning-program-rtyr7/projects/googlecloud-form-parsing-with-document-ai-python-zdlse***		punctuation. The output should be a list of (token, token_type) pairs. Implement an algorithm that takes a given regular expression (e.g., (a|b)*c) and constructs its equivalent Non-deterministic Finite Automaton (NFA). You can represent the NFA using adjacency lists or a similar data structure. Write a program that takes an NFA (represented as in the previous question) as input and converts it into	

					an equivalent Deterministic Finite Automaton (DFA) using the subset construction algorithm. Design a simple programming language with a few keywords, operators, and data types. Then, implement a handwritten lexical analyzer (without using tools like LEX) in a language like Python or Java that can tokenize programs written in this small language. Write a LEX program (or Flex, its open-source	

					counterpart) that can identify and classify the following tokens in an input file: keywords (if, else, while, int), identifiers (starting with a letter, followed by letters or digits), integer literals, and basic arithmetic operators (+, -, *, /). Extend the LEX program from the previous question to ignore C-style (/* ... */) and C++-style (// ...) comments, as well as whitespace (spaces, tabs, newlines). The output should only contain the	

					identified tokens. Given a context-free grammar for a simple arithmetic expression language (e.g., with addition, subtraction, multiplication, division, and parentheses), write a program that can: Determine if the grammar is ambiguous. (Hint: You might try to derive the same string in two different ways). Draw the parse tree for a given valid expression	

						according to the grammar. Write a program that takes a context-free grammar as input and removes any immediate left recursion present in its production rules. The output should be the modified grammar without left recursion. Choose either a top-down parsing technique (like recursive descent) or a bottom-up technique (like shift-reduce) and implement a parser for a small subset of a programming	

					language based on a given context-free grammar. The parser should take a sequence of tokens as input and output whether the input is syntactically valid according to the grammar. For a top-down parser, you might need to handle LL(1) conditions. Research and prepare a report or a short presentation explaining some common non-context-free constructs found in programming languages (e.g., declaration before use, type checking	

					consistency). Provide examples of how these constructs cannot be easily handled by standard context-free grammars.	
2	**Top-down parsing**	FIRST & FOLLOW sets, LL(1) conditions, predictive parsing, recursive descent parsing, error recovery. LR-parsing - Handle pruning, shift-reduce parsing, viable prefixes, valid items, LR(0) automaton, LR-parsing algorithm, SLR(1), LR(1), and LALR(1) parsing. YACC, error recovery with YACC and examples of YACC specifications. Syntax-directed definitions (attribute grammars)-Synthesized and inherited attributes, examples of SDDs, evaluation orders for attributes of an SDD, dependency graphs. S-attributed and L- attributed SDDs and their implementation using LR-parsers and recursive	***International Academia:*** ***https://ocw.mit.edu/courses/6-004-computation-structures-spring-2017/pages/c11/*** (MIT Open Courseware): ***AICTE-prescribed syllabus:*** ***https://drive.google.com/file/d/11CBeslrHOERE-XHDJSNsIHlRONmt_tYX/view?usp=sharing*** ***Industry Mapping:***	10	Write a program that takes a context-free grammar (CFG) as input and computes the FIRST and FOLLOW sets for all non-terminals in the grammar. The output should clearly display the grammar and the calculated FIRST and FOLLOW sets. Test your program with various grammars, including those with epsilon productions and	**Textbook-1** Chapters: 4, 5 **Page No.:** 159 - 341

		descent parsers respectively.	***https://www.coursera.org/programs/iem-faculty-learning-program-rtyr7/projects/googlecloud-form-parsing-with-document-ai-python-zdlse***		left recursion. Develop a program that takes a CFG and the computed FIRST and FOLLOW sets as input. The program should then check if the given grammar satisfies the LL(1) conditions. The output should indicate whether the grammar is LL(1) or not, and if not, identify the violating productions. Implement an algorithm that takes an LL(1) grammar (or one that can be made LL(1) by left-factoring) and constructs its	

					predictive parsing table. The output should be a clear representation of the parsing table. Write a program that implements a non-recursive predictive parser using the parsing table generated in the previous question. The parser should take a string as input and determine if it is accepted by the grammar. The output should show the sequence of moves made by the parser (stack, input, action). Include basic error detection for	

					invalid input strings. Design a simple context-free grammar (e.g., for arithmetic expressions with basic operators). Implement a recursive descent parser for this grammar in your chosen programming language. The parser should take an input string and output whether it is a valid sentence of the grammar. Write a program that simulates the actions of a shift-reduce parser for a given grammar and input string. The program should output	

					the sequence of stack contents, input remaining, and the action (shift or reduce) performed at each step. Include the detection of acceptance and error states. Implement an algorithm that takes a context-free grammar and constructs its LR(0) automaton (the collection of LR(0) items and the transitions between them). The output should clearly represent the states and transitions of the automaton. Extend the LR(0)	

					automaton construction to generate the SLR(1) parsing table. Your program should take a grammar and the LR(0) automaton as input and produce the SLR(1) parsing table (action and goto parts). Handle potential conflicts and indicate if the grammar is SLR(1). Write a YACC (or Bison) specification for a simple programming language construct (e.g., assignment statements, simple arithmetic expressions, or a	

					basic conditional statement). Compile and test your YACC specification with valid and invalid input. Observe how YACC handles the parsing and reports syntax errors. Consider a simple syntax-directed definition (SDD) for calculating the value of arithmetic expressions. Implement a program that takes a parse tree (you can manually construct simple parse trees for testing) and evaluates the	

					attributes according to the SDD rules. Demonstrate the evaluation process for both synthesized and inherited attributes (if applicable in your chosen SDD).	
3	**Semantic analysis**	Symbol tables and their data structures. Representation of “scope”. Semantic analysis of expressions, assignment, and control-flow statements, declarations of variables and functions, function calls, etc., using S- and L-attributed SDDs (treatment of arrays and structures included). Semantic error recovery Intermediate code generation - Different intermediate representations –quadruples, triples, trees, flow graphs, SSA forms, and their uses. Translation of expressions (including array references with subscripts) and	***International Academia:*** ***https://ocw.mit.edu/courses/6-004-computation-structures-spring-2017/pages/c11/*** (MIT Open Courseware): ***AICTE-prescribed syllabus:*** ***https://drive.google.com/file/d/11CBeslrHOERE-XHDJSNsIHlRON***	10	Create functions to insert, lookup, and delete entries in the symbol table. Each entry should store at least the variable name and its data type. Demonstrate the insertion and retrieval of symbol information for a small set of declared variables. Modify the	**Textbook-1** Chapters: 5, 6 **Page No.:** 279 to 388

		assignment statements. Translation of control-flow statements – it-then- else, while-do, and switch. Short-circuit code and control-flow translation of Boolean expressions. Back patching. Examples to illustrate intermediate code generation for all constructs	***mt_tYX/view?usp=sharing*** ***Industry Mapping:*** ***https://www.coursera.org/programs/iem-faculty-learning-program-rtyr7/projects/googlecloud-form-parsing-with-document-ai-python-zdlse***		symbol table implementation to support the concept of scope. Implement a mechanism to enter and exit scopes. Demonstrate how variables declared in different scopes are managed and how name resolution works (e.g., accessing a variable in an outer scope). Write a program that takes simple expressions (e.g., a + b, x < y) as input. Perform type checking to ensure that the operands are compatible. Report any type errors (e.g.,	

					adding an integer to a boolean). Assume variables have been previously declared and their types are available in the symbol table. Write a program that takes assignment statements (e.g., x = y + 5) as input. Perform type compatibility checks between the left-hand side variable and the right-hand side expression. Handle implicit type conversions (if applicable in your chosen scope) and report type errors.	

					Write a program that takes if-then-else statements as input. Ensure that the condition expression is of a boolean type. Demonstrate the semantic analysis process. Write a program that processes variable and function declarations. For variables, store their names and types in the symbol table. For functions, store their names, return types, and the types and number of their parameters. Detect and report redeclaration	

					errors. Write a program that takes function calls as input. Check if the function being called has been declared, if the number of arguments matches the function definition, and if the types of the actual arguments are compatible with the formal parameters. Report any errors. Write a program that takes arithmetic expressions as input and generates their corresponding quadruple representation.	

					Each quadruple should have the format (operator, operand1, operand2, result). Handle operator precedence and associativity appropriately. Write a program that takes assignment statements and while-do loops as input and generates their corresponding triple representation. Each triple should have the format (operator, operand1, operand2). For control flow, you'll need to represent the conditions and	

					jump targets using triple indices. Take a simple sequence of assignments and arithmetic operations with variable reassignments. Convert this code into SSA form by introducing new versions of variables where necessary. Demonstrate the key property of SSA: each variable is assigned a value only once.	
4	**Run-time environments**	Stack allocation of space and activation records. Access to non-local data on the stack in the case of procedures with and without nesting of procedures. Introduction to machine code generation and optimization- Simple	***International Academia:*** [***https://ocw.mit.edu/courses/6-004-computation-structures-spring-2017/pages/c11/***](https://ocw.mit.edu/courses/6-004-computation-structures-spring-2017/pages/c11/)	8	Design the layout of an activation record for a function in a hypothetical programming language. Include space for parameters,	**Textbook-1** **Chapters:** 7, 8, 9, 10 **Page No.:** 389 to 751

		machine code generation, examples of machine-independent code optimizations.	(MIT Open Courseware): ***AICTE-prescribed syllabus:*** ***https://drive.google.com/file/d/11CBeslrHOERE-XHDJSNsIHlRONmt_tYX/view?usp=sharing*** ***Industry Mapping:*** ***https://www.coursera.org/programs/iem-faculty-learning-program-rtyr7/projects/googlecloud-form-parsing-with-document-ai-python-zdlse***		local variables, the return address, and the control link (dynamic link). Illustrate how this layout would be instantiated for a simple function call with a few parameters and local variables. Write a program (in any language) to simulate the stack during a sequence of simple function calls (without nesting). The program should track the stack pointer and the contents of each activation record pushed onto and popped from the stack. Demonstrate	

					this with at least three sequential function calls. Write a small program with a function that declares and uses local variables. Then, manually trace the execution, showing how the activation record for this function is created on the stack and how the local variables are accessed (relative to the frame pointer). Implement a function that takes parameters passed by value. Simulate the stack during a call to this function,	

					showing how the parameter values are copied into the activation record. Implement a function that takes parameters passed by reference (if your chosen simulation language allows). Simulate the stack during a call, illustrating how the addresses of the actual arguments are stored in the activation record. Extend the stack simulation from question 2 to include nested function calls.	

					Demonstrate how the control links in the activation records are used to maintain the dynamic chain and enable the correct return from nested calls. Include at least one level of nesting. Consider a program with nested procedures and static scoping. Implement a scenario where an inner procedure needs to access a variable declared in an enclosing procedure. Manually trace the stack and demonstrate how the static	

					link (access link) in the activation record of the inner procedure would be used to find the non-local variable in the activation record of the outer procedure. Choose a very simple arithmetic expression (e.g., a = b + c * 2). Design a sequence of hypothetical machine code instructions (e.g., using a simplified assembly-like language with instructions like LOAD, STORE, ADD, MUL) to evaluate this expression. Assume	

					variables a, b, and c are stored in memory locations. Take a code snippet with constant expressions (e.g., x = 5 + 3 * 2; y = z + 8). Apply the constant folding optimization manually to simplify the code. Show the original and optimized code snippets. Write a small code segment that includes a variable that is assigned a value but never subsequently used. Identify this dead code and demonstrate how it can be	

					eliminated to optimize the code. Show the original and optimized code snippets.	

Textbooks:

1. Compilers: Principles, Techniques, and Tools, by A.V. Aho, Monica Lam, Ravi Sethi, and J.D. Ullman, (2nd ed.), Addison-Wesley, 2007 (main text book, referred to as ALSU in lab assignments).

Reference books:

1. K.C. Louden, Compiler Construction: Principles and Practice, Cengage Learning,
2. D. Brown, J. Levine, and T. Mason, LEX and YACC, O?Reilly Media, 1992.

Online Resources: https://drive.google.com/file/d/19i8gYQD2xDzf8nUCxdqp7czFLhaDrVFp/view?usp=sharing

CO-PO Mapping:

	PO1	**PO2**	**PO3**	**PO4**	**PO5**	**PO6**	**PO7**	**PO8**	**PO9**	**PO10**	**PO11**	**PO12**
CO1	3	3		3	3				3	3		3
CO2	3	2		3	3						2	2
CO3	3	2		3	3				3	2		3

CO4	3	2	1	2	2				2	2		2

3: Strong correlation
2: Medium correlation
1: Weak correlation

University of Engineering and Management

Institute of Engineering & Management, Salt Lake Campus

Institute of Engineering & Management, New Town Campus

University of Engineering & Management, Jaipur

Department of Computer Science

B.Tech in CSE (AI & ML)

DETAILED SYLLABUS

Course Code - PECCS701D
Course Title - Computer Vision
Credit – 3
Category – Professional Elective
Semester – VII
L:T:P:S – 3:0:0:0
Prerequisites - Familiarity with Python programming, including NumPy, OpenCV basics, and linear algebra fundamentals.

Course Outcomes:

CO1	Understand and apply basic image processing techniques such as filtering, geometric transformations, and histogram operations.
CO2	Implement and evaluate feature detection and matching techniques for object recognition and image registration.
CO3	Apply segmentation algorithms and contour analysis methods for object extraction and shape analysis in images.
CO4	Design and train convolutional neural networks (CNNs) for image classification tasks and apply object detection models such as YOLO/SSD.
CO5	Demonstrate practical skills in using popular computer vision libraries (OpenCV, Keras, TensorFlow) to develop end-to-end vision pipelines.

Study Material	Coursera	NPTEL	Linkedin Learning	Infosys Springboard 5G

Module No	Topic	Sub Topic	Mapping with Industry and International Academia	Contact Hours	Assignment Question Mapped with the Module	Reference Textbooks

1	**Image Processing Fundamentals**	Introduction to Computer Vision Image Representation: Grayscale, RGB, Binary Basic Image Processing Operations: Filtering, Blurring, Sharpening Geometric Transformations: Scaling, Rotation, Translation Image Histograms and Histogram Equalization	***NPTEL:*** Fundamentals of Digital Image Processing: https://onlinecourses.nptel.ac.in/noc22_ee116/preview?utm_source=chatgpt.com ***International Academia:*** ***(MIT OCW)*** Computational Photography https://www.youtube.com/playlist?list=PLUl4u3cNGP61pwA6paIRZ30q1sjLE8b6c ***Industry Mapping:*** Image Processing, Robotic Vision, AR/VR	9	1.Implement a Python program to apply Gaussian blur, median filter, and sharpening on an image. Compare the results. 2. What is the difference between image blurring and sharpening? Explain with suitable filters.	**Text Book:** Chapter 1 Page 1-44

2:	**Feature Detection and Matching**	Edge Detection: Sobel, Canny Corner Detection: Harris Corner Detector Keypoint Detection and Description: SIFT, SURF, ORB Feature Matching: Brute-Force and FLANN based matcher	***International Academia:*** (MIT OpenCV tutorials) https://fab.cba.mit.edu/classes/865.15/people/desmond.lim/project-06.html ***Industry Mapping***: Autonomous Vehicle, AR/VR, Robotics	9	1. Apply Canny edge detection on an image and visualize the edges. 2. Detects corners in an image using the Harris Corner Detector.	**Text Book 3:** Chapter 4
3	**Image Segmentation and**	Image Thresholding:	***NPTEL***	9	1.	**Text Book 1:**

	Contour Analysis	Global, Adaptive Contour Detection and Analysis Morphological Operations Image Segmentation: Watershed, GrabCut	(Computer Vision) https://onlinecourses.nptel.ac.in/noc21_cs101/preview?utm_source=chatgpt.com ***Industry Mapping:*** Used in biomedical imaging, satellite image analysis, surveillance		Perform adaptive thresholding on an image with uneven illumination. 2. Detect contours in an image and calculate area, perimeter, and bounding box for each contour.	Chapter 10
4	**Deep Learning for Computer Vision**	Introduction to Convolutional Neural Networks (CNNs) CNN Architectures: LeNet, AlexNet, VGG	***NPTEL*** (Deep Learning)	9	1. Build and train a simple CNN for CIFAR-10 or MNIST image classification. Q2. Apply transfer learning using a pretrained VGG16	**Text Book 2:** Chapter 1 - 4

		Image Classification using CNNs Object Detection: YOLO, SSD basics Transfer Learning for Vision Tasks	https://onlinecourses.nptel.ac.in/noc21_cs101/preview?utm_source=chatgpt.com ***Industry Mapping:*** Essential for ML Engineer, AI Researcher, and Data Scientist roles		model to classify custom image dataset (flowers, animals, etc.).	

Textbooks:

1. "Digital Image Processing" by Rafael C. Gonzalez and Richard E. Woods, Pearson

2. Deep Learning for Computer Vision" by Rajalingappaa Shanmugamani

3. Computer Vision: Algorithms and Applications" by Richard Szeliski

Reference books:

1. "Learning OpenCV 4" by Gary Bradski and Adrian

2. Programming Computer Vision with Python" by Jan Erik Solem

CO-PO Mapping:

	PO1	**PO2**	**PO3**	**PO4**	**PO5**	**PO6**	**PO7**	**PO8**	**PO9**	**PO10**	**PO11**	**PO12**
CO1	3	2	2	-	2	-	-	-	-	-	-	2
CO2	3	3	3	2	-	-	-	-	2	-	-	2
CO3	3	2	3	2	-	-	-	-	-	-	-	2
CO4	3	3	3	3	3	-	-	-	2	-	2	3

3: Strong correlation

2: Medium correlation

1: Weak correlation

University of Engineering & Management, Kolkata
University of Engineering & Management, Jaipur
Institute of Engineering & Management, Kolkata
Department of Computer Science

DETAILED SYLLABUS

Course Code- PECCS701L
Course Title – Advanced Deep Learning
Credit – 3
Category – Professional Elective Course
Semester – VII
L:T:P:S – 3:0:0:0
Pre-requisite – Basic understanding of Machine Learning and Neural Networks

Course Outcomes:

CO1	Explain the fundamentals of concepts of deep learning.
CO2	Analyse the dynamics of deep learning networks.
CO3	Develop Deep Learning based solutions for modern problems.
CO4	Conduct advanced research and analysis on various frontiers of deep learning.

Study Material	Coursera	NPTEL	Linkedin Learning	Infosys Springboard 5G
Link				

Module No.	Topic	Sub-topics	Mapping with Industry and International Academia	Lecture Hours	Corresponding Lab Assignments	Textbook Mapping
1	Introduction to Artificial Neural Networks	Perceptron, Activation & Loss Functions, SGD, MLP, Backpropagation, Autoencoders	Coursera (Deep Learning *https://www.coursera.org/specializations/deep-learning* Specialization), NPTEL *https://onlinecourses.nptel.ac.in/noc21_cs76/preview*	9	Implementing Perceptron, MLP, Autoencoder	Goodfellow et al. Ch 6, 7
2	Controlling Neural Networks	Momentum, Adaptive Learning Rates, Hyperparameter Tuning, Regularization, Gradient Issues	Coursera, NPTEL	7	Experimenting with Adam, RMSProp, Dropout	Goodfellow et al. Ch 8

3	Deep Learning Architectures	CNN, RNN, Encoder-Decoder, GANs, Transformers	LinkedIn Learning, Stanford NLP Lectures ***https://web.stanford.edu/class/cs224n/***	7	CNN on CIFAR-10, LSTM for Text, GAN for MNIST	Goodfellow et al. Ch 9, 10
4	Deep Learning Models	AlexNet, VGGNet, Inception, ResNet, SegNet, U-Net, LSTM, GRU, GAN, CycleGAN, BERT, GPT	Academic papers, OpenAI GPT, Google BERT, PyTorch Docs ***https://www.coursera.org/learn/advanced-deep-learning-with-pytorch?utm_source=chatgpt.com***	7	Fine-tune VGG, U-Net segmentation, text gen using GPT	Reference Papers, Goodfellow et al.
5	Advanced Techniques	Transfer Learning, Reinforcement Learning, Evolutionary Networks, Federated & Active Learning, XAI	Infosys Springboard, Google AI Blog, OpenAI Gym ***https://biods220.stanford.edu/lectures/lecture14.pdf*** ***https://www.infosys.com/about/springboard.html***	8	Transfer Learning for Flowers, RL with Gym, XAI tools	Campesato, Online Papers

Textbooks:

1. Deep Learning by Ian Goodfellow, Yoshua Bengio, Aaron Courville

2. Artificial Intelligence, Machine Learning and Deep Learning by Oswald Campesato

Reference books:

Selected research papers on GANs, Transformers, Federated Learning

https://arxiv.org/pdf/2106.11342

Online Resources:

- Coursera Deep Learning Specialization (Andrew Ng)
- Stanford CS231n, CS224n
- PyTorch and TensorFlow Documentation

CO-PO Mapping:

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
CO1	3	3	-	-	-	-	-	-	-	-	-	-
CO2	2	3	3	-	-	-	-	-	-	-	-	-
CO3	-	2	3	3	2	-	-	-	-	-	-	-
CO4	-	-	3	3	-	-	-	-	-	3	-	-

3: Strong correlation

2: Medium correlation

1: Weak correlation

University of Engineering & Management, Kolkata
University of Engineering & Management, Jaipur
Institute of Engineering & Management, Kolkata
Department of Computer Science

B.Tech in CSE (AI & ML)

DETAILED SYLLABUS

Course Code- OECCS701A
Course Title – Enterprise Systems
Category – Open Elective Course
Credit – 3
Semester – VII
L:T:P:S – 3:0:0:0
Pre-requisite – Basic knowledge of Business, Finance & Accounting, Awareness on Business Environment

Course Outcomes:

CO1	**Understand Enterprise systems models.**

CO2	**Understand Enterprise Software architecture.**
CO3	**Design and implement ERP models.**
CO4	**Implement interactive networks and applications.**
CO5	**Develop models for ERP for large projects.**

Module No.	Topic	Sub-topics	Mapping with Industry and International Academia	Lecture Hours	Corresponding Lab Assignments	Textbook Mapping
1	**Introduction to Enterprise systems concepts**	Features, capabilities and Overview of Commercial Software, re-engineering work processes for IT applications, Business Process Redesign, Knowledge engineering and data warehouse. Business Modules: Finance, Manufacturing (Production), Human Resources, Plant Maintenance, Materials Management, Quality Management, Sales & Distribution	***International Academia:*** (Lecture Notes | Logistics and Supply Chain Management | Engineering Systems Division | MIT OpenCourseWare) ***AICTE-prescribed syllabus:*** (Final BTECH LSCM.pdf) ***Industry Mapping:***	6	1. Familiarization with ERPNext - open-source alternative to SAP. 2. Familiarization with Odoo ERP software. 3. Familiarization with anyLogistix software.	**Textbook-1** Chapters: 1, 2 and 3 **Textbook-2** Chapters: 1-4, 11 and part V.

			ERPNext, anyLogistix, Odoo (***GitHub - frappe/erpnext: Free and Open Source Enterprise Resource Planning (ERP)***,Case study – anyLogistix, Open Source ERP and CRM | Odoo)			
2	**Enterprise Resource Planning (ERP)Architecture and Technologies**	Service Oriented Architecture (SOA): Principles of loose coupling, encapsulation, Interoperability. Decision Support System: On-Line Analytical Processing, Electronic Data Exchange, Customer Relationship Management (CRM), Supplier Relationship Management (SRM),	***International Academia:*** (Lecture Notes | Logistics and Supply Chain Management | Engineering Systems Division | MIT OpenCourseWare) ***AICTE-prescribed syllabus:*** (Final BTECH LSCM.pdf) ***Industry Mapping:***	8	1. Create a model of customer relationship management and business intelligence systems for catalogue and online retailers. 2. Create a model of Supplier Relationship Management for Healthcare system.	**Textbook-1** Chapter: 3 **Textbook-2** Chapter: part II.

			ERPNext, anyLogistix, Odoo (***GitHub - frappe/erpnext: Free and Open Source Enterprise Resource Planning (ERP)***,Case study – anyLogistix, Open Source ERP and CRM | Odoo)			
3	**ERP Software Architecture and Technologies**	Introduction to MVC, MVC method of software development in a 3-tier environment, Microsoft .NET framework, PHP, Ruby on Rails, JavaScript, Ajax and Overview of SAP and Oracle Applications		10	1. Create ASP.NET MVC Application using Visual Studio. 2. Create JSP application using MVC framework	**Text Book-3** Chapters:1 - 6. **Coursera course: ASP.NET Core MVC [.NET 8] - The Complete Guide Specialization** ***https://www.coursera.org/specializations/packt-asp-net-core-mvc-net-***

						8-the-complete-guide
4	**ERP Network Architecture**	Introduction to MPLS, Virtual Private Networks (VPN), Storage area networks, Storage units, Back-up strategies, Local Area Network (LAN) technologies and products, Data Centres. Firewalls, Network monitoring and enforcement of policies, ERP Security Issues, Authentication, Authorisation, Access control, Roles, single-sign on, Directory servers, Audit trails, Digital signatures, Encryption, review of IPSec, SSL.	***International Academia:*** *(1.264J Lecture 37 Notes: Networks: Enterprise, VPN, MPLS. Course summary, Lecture Notes | Logistics and Supply Chain Management | Engineering Systems Division | MIT OpenCourseWare)* ***AICTE-prescribed syllabus:*** *(Final BTECH LSCM.pdf)* ***Industry Mapping: ERPNext, anyLogistix, Odoo (GitHub - frappe/er***	10	1. Configure and test a VPN connection. 2. Configure Firewall in Windows/Linux OS. 3. Configure an MPLS VPN in Cisco Packet Tracer.	**Text Book-4** Chapters: 1-12. **Enterprise Network Infrastructure : Coursera course** ***https://www.coursera.org/learn/packt-enterprise-network-infrastructure-pbclr?msockid=3e887050530967811ce4656152bb66d3***

			pnext: Free and Open Source Enterprise Resource Planning (ERP), *Case study – anyLogistix, Open Source ERP and CRM | Odoo)*			
5	**Trends in ERP**	Enterprise Application Integration (EAI), ERP and E-Commerce, ERP and Internet, Future Directions in ERP	***International Academia:*** (Lecture Notes | Logistics and Supply Chain Management | Engineering Systems Division | MIT OpenCourseWare) ***AICTE-prescribed syllabus:*** (Final	6	Design with the help of CASE tools to aid ERP Software acquisition process - Case study	**Text Book-1** Chapter 11

			BTECH LSCM.pdf) ***Industry Mapping: ERPNext, anyLogistix, Odoo*** ***(GitHub - frappe/erpnext: Free and Open Source Enterprise Resource Planning (ERP)***,Case study – anyLogistix, Open Source ERP and CRM | Odoo)			

Textbooks:

1. Alexis Leon, Enterprise Resource Planning, 2020,4th Edition, Tata McGraw Hill.
2. Alexis Leon, ERP Demystified, McGrawhill education.
3. Simone Chiaretta, Keyvan Nayyeri, Beginning ASP.NET MVC 1.0, Wrox Press.
4. A Practical Introduction to Enterprise Network and Security Management, Bongsik Shin, CRC Press Taylor & Francis Group.

Reference books:

1. The Age of Metapreneurship: A Journey into the Future of Entrepreneurship Design of Enterprise Systems: Theory, Architecture, and Methods, Ronald Giachetti.
2. Entrepreneurship: The Practice and Mindset Paperback, by Heidi M. Neck (Author), Christopher P. Neck (Author), Emma L. Murray.

3. ENTREPRENEURSHIP: The Art, Science, and Process for Success, Charles E. Bamford Associate Professor of Strategy & Entrepreneurship (Author), Garry D.Bruton Dr.

Online Resources:

Coursera course:

1. Enterprise Systems, Jason Chan, University of Minnesota.

LinkedIn course:

1. Enterprise Architecture Foundations, Dave Swersky.

CO-PO Mapping:

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
CO1	3	2	1	1	-	-	-	-	-	-	-	-
CO2	3	2	1	1	-	-	-	-	-	-	-	-
CO3	3	2	2	2	2	-	-	-	-	-	-	-
CO4	3	2	2	2	2	-	-	-	-	-	-	-
CO5	3	3	2	2	2	-	-	-	-	-	-	2

3: Strong correlation

2: Medium correlation

1: Weak correlation

University of Engineering & Management, Kolkata
University of Engineering & Management, Jaipur
Institute of Engineering & Management, Kolkata

Department of Computer Science

B.Tech in CSE (AI & ML)

DETAILED SYLLABUS

Course Code- OECCS701C
Course Title – Entrepreneurship and Startups
Credit – 3
Semester – VII
L:T:P:S – 3:0:0:0
Pre-requisite – Basic knowledge of Business, Finance & Accounting, Awareness on Business Environment

Course Outcomes:

CO1	Remembering the basic concept of Entrepreneurship.
CO2	Understanding and applying different skills of Entrepreneurship.
CO3	Analyze various types of start-up ecosystems and strategies for implementing and its uses in the present market.

CO4	Understanding of trends, opportunities and financial evaluation of Start Up Opportunities and Create new concepts to overcome Modern Entrepreneurship challenges.

Study Material	Coursera	NPTEL	LinkedIn learning.	

Module No.	**Topic**	**Sub-topics**	**Mapping with Industry and International Academia**	**Lecture Hours**	**Corresponding Lab Assignments**	**Textbook Mapping**
1	Introduction to Entrepreneurship	Meaning and concept of entrepreneurship, the history of entrepreneurship development, role of entrepreneurship in economic development, Myths about entrepreneurs, agencies in entrepreneurship management future of entrepreneurship types of entrepreneurs	***International Academia:*** (MIT Open Courseware): *https://ocw.mit.edu/collections/entrepreneurship/* ***AICTE-prescribed syllabus:*** https://www.aicte-india.org/sites/default/files/Awarene	8		**Textbook-1** Chapters: 1 to 8 **Textbook-2** Chapters: page no. 1 to 192

			ss%20Workshop_Vocational.pdf ***Industry Mapping: NA***			
2	The Entrepreneur	Why to become entrepreneur, the skills/ traits required to be an entrepreneur, Creative and Design Thinking, the entrepreneurial decision process, skill gap analysis, and role models, mentors and support system, entrepreneurial success stories. Communication: Importance of communication, barriers and gateways to communication, listening to people, the power of talk, personal selling, risk taking & resilience, negotiation.	***International Academia:*** (MIT Open Courseware): *https://ocw.mit.edu/collections/entrepreneurship/* ***AICTE-prescribed syllabus:*** https://www.aicte-india.org/sites/default/files/Awareness%20Workshop_Vocational.pdf ***Industry Mapping: NA***	8		**Textbook-1** Chapters: 9, 17 **Textbook-2** Chapters: page no. 193 to 354

3	Start-up Introduction	various form of business organization (sole proprietorship, partnership, corporations, Limited Liability,) Company mission, vision and strategy formulation.	***International Academia:*** (MIT Open Courseware): https://ocw.mit.edu/MIT Courses/entrepreneurship/ ***AICTE-prescribed syllabus:*** https://www.aicte-india.org/sites/default/files/Awareness%20Workshop_Vocational.pdf ***Industry Mapping:*** *NA*	14	NA	**Textbook-1** Chapters: 18 to 24 **Textbook-2** Chapters: page no. 355 to 610
4	Finance and Accounting concepts	Rate of Return Methods, Break even analysis, Financial statements, basic accounting concept. .	***International Academia:*** (MIT Open Courseware):	6	NA	**Textbook-1** Chapters: 25 to 29 **Textbook-2** Chapters: 611

		E-Cell: Meaning and concept of E-cells, advantages to join E-cell, significance of E-cell, various activities conducted by E-cell.	***https://ocw.mit.edu/collections/entrepreneurship/*** ***AICTE-prescribed syllabus:*** https://www.aicte-india.org/sites/default/files/Awareness%20Workshop_Vocational.pdf ***Industry Mapping: NA***			to 762

Textbooks:

1. "Entrepreneurship Development and Business Ethics" by Mukherjee & Roy (Oxford University Press)
2. "Entrepreneurship Development and Business Ethics" by Desai V (Himalaya Publishing House)

Reference books:

1. The Age of Metapreneurship: A Journey into the Future of Entrepreneurship Design of Enterprise Systems: Theory, Architecture, and Methods, Ronald Giachetti.
2. Entrepreneurship: The Practice and Mindset Paperback, by Heidi M. Neck (Author), Christopher P. Neck (Author), Emma L. Murray.
3. ENTREPRENEURSHIP: The Art, Science, and Process for Success, Charles E. Bamford Associate Professor of Strategy & Entrepreneurship (Author), Garry D.Bruton Dr.

Online Resources:

CO-PO Mapping:

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
CO1												
CO2												
CO3												
CO4												

University of Engineering & Management, Kolkata

University of Engineering & Management, Jaipur

Institute of Engineering & Management, Kolkata

Department of Computer Science

B.Tech in CSE (AI & ML)

DETAILED SYLLABUS

Course Code- ESP(CS)701

Course Title – Essential Studies for Professionals (CS) - VII

Credit – 0.5

Category – Humanities & Social Sciences including Management course

Semester – VII

L:T:P:S – 2:0:0:0

Pre-requisite – Basic knowledge of Data Structures, Basics of Compiler, Operating systems and Computer network

Course Outcomes:

CO1	To develop a detailed knowledge of compiler designs.
CO2	To learn all types of Data Base Management Systems' fundamentals.
CO3	To understand Operating Systems and its applications.
CO4	To use fundamentals of Computer networks and its methods.

Module No.	Topic	Sub-topics	Mapping with Industry and International Academia	Lecture Hours	Textbook Mapping
1	**Compiler Design**	Lexical analysis, parsing, syntax-directed translation. Runtime environments. Intermediate code generation. Local optimization, Data flow analyses: constant propagation, likeness analysis, common sub expression elimination	***International Academia:*** (MIT Open Courseware): ***AICTE-prescribed syllabus:*** ***Industry Mapping:***	6	1. G.K publishers GATE Computer Science Engineering, 2. McGraw hill GATE 2020 Computer Science Engineering, 3. Wiley GATE 2020 Computer Science Engineering
2	**Databases**	Integrity constraints, normal forms. File organization, indexing (e.g., B and B+ trees). Transactions and concurrency control	***International Academia:*** (MIT Open Courseware): ***AICTE-prescribed syllabus:*** ***Industry Mapping:***	12	1. G.K publishers GATE Computer Science Engineering, 2. McGraw hill GATE 2020 Computer Science Engineering, 3. Wiley GATE 2020 Computer Science Engineering
3	**Operating System**	Processes, threads, inter-process communication, concurrency and synchronization, Deadlock, CPU scheduling. Memory management and virtual memory, File system. Flow and error control techniques, switching. IPv4/IPv6, routers and routing algorithms (distance vector, link state). TCP/UDP and sockets, congestion control.	***International Academia:*** (MIT Open Courseware): ***AICTE-prescribed syllabus:***	12	1. G.K publishers GATE Computer Science Engineering, 2. McGraw hill GATE 2020 Computer Science Engineering, 3. Wiley GATE 2020 Computer Science Engineering

		Application layer protocols (DNS, SMTP, POP, FTP, HTTP). Basics of Wi-Fi.	***Industry Mapping:***		
4	**Computer Networks**	Concept of layering: OSI and TCP/IP Protocol Stacks; Basics of packet, circuit and virtual circuit- switching; Data link layer: framing, error detection, Medium Access Control, Ethernet bridging; Routing protocols: shortest path, flooding, distance vector and link state routing; Fragmentation and IP addressing, IPv4, CIDR notation, Basics of IP support protocols (ARP, DHCP, ICMP), Network Address Translation (NAT); Transport layer: flow control and congestion control, UDP, TCP, sockets; Application layer protocols: DNS, SMTP, HTTP, FTP, Email	***International Academia:*** (MIT Open Courseware): ***AICTE-prescribed syllabus:*** ***Industry Mapping:***	6	1. G.K publishers GATE Computer Science Engineering, 2. McGraw hill GATE 2020 Computer Science Engineering, 3. Wiley GATE 2020 Computer Science Engineering

Text books:

1. G.K publishers GATE Computer Science Engineering,
2. McGraw hill GATE 2020 Computer Science Engineering,
3. Wiley GATE 2020 Computer Science Engineering

Online Resources:

Module	Platform	Course Link
Compiler Design	NPTEL	Link
Compiler Design	Stanford	https://youtu.be/sm0QQO-WZlM
Databases	NPTEL	Link
Databases	Coursera (IBM)	Link
Operating Systems	NPTEL	Link
Operating Systems	Coursera (Google)	Link
Computer Networks	NPTEL	Link
Computer Networks	Coursera (Stanford)	Link

CO-PO Mapping:

CO \ PO	PO1 (Engineering Knowledge)	PO2 (Problem Analysis)	PO3 (Design/Dev of Solutions)	PO4 (Investigation)	PO5 (Modern Tool Usage)	PO6 (Engineer & Society)	PO7 (Environment & Sustainability)	PO8 (Ethics)	PO9 (Individual & Team Work)	PO10 (Communication)	PO11 (Project Management)	PO12 (Life-long Learning)
CO1	2	1	1	1	1	-	-	-	-	-	-	3
CO2	2	1	1	1	1	-	-	-	1	1	1	3
CO3	2	1	1	1	1	1	-	-	1	1	1	3
CO4	2	1	1	1	1	1	-	-	1	1	1	3

3: Strong correlation

2: Medium correlation

1: Weak correlation